A CONJECTURE FROM LEARNING SIMULATIONS OF SERIES AND PARALLEL CONNECTIONS OF COMPONENTS

Alexandre Muzy, Bernard P. Zeigler

A CONJECTURE FROM LEARNING SIMULATIONS OF SERIES AND PARALLEL CONNECTIONS OF COMPONENTS

Alexandre Muzy*, Bernard P. Zeigler‡

*I3S UMR CNRS 7271, Bio-info, CS 40121 - 06903 Sophia-Antipolis Cedex, France, Email: alexandre.muzy@cnrs.fr.

‡RTSync Corp. and Arizona Center for Integrative Modeling and Simulation, AZ, United-States of America, Email: zeigler@rtsync.com.

ABSTRACT

Abstract sequential machines can be realized by series and parallel connections of components. These connections can be automatically learned by simulation, rating and selecting component models. Our goal here is threefold: (i) discuss implications for model reconstruction and reconfiguration in model engineering, (ii) present mathematical analyses of search simulation results, and (iii) conjecture that the whole search space of achievement components seems to be well ordered. This conjecture opens new perspectives for studying parallel and series connections of components upon a simulation-based framework. More generally, the automatic learning demonstrated here constitutes a formal basis for model reconstruction and reconfiguration in model engineering.

Keywords: Sequential machines, activity, discrete-events, simulation, learning.

1 INTRODUCTION

Linear sequential machines can be decomposed into parallel and series connections [1, 2]. In sequential machine theory [3], logical and functional functions in machines are studied describing possible realizations independently from the implementation of physical components.

On the other hand, *activity* can be used to link the energy of physical components to information [4] and activity-based credit assignment (ACA) can be used for automatically rating component systems and composing them according to the experimental frames in which they are placed [5, 6]. This includes both arbitrarily connected coupled models, i.e., having series, parallel, or feedback loop couplings in any combination. Here we focus on series and parallel coupled models where mathematical analysis of the ACA's behavior is possible, as we shall show.

From a systems science perspective, *systems problem solving* [7] and *system design and engineering* [8] are the approaches closest to the approach presented here. However, these approaches remain at modeling level and do not use activity at component level for building systems at network level. Concerning other credit assignment methods, a comparison is presented in [5].

This manuscript aims at using simulation for inductively modeling the automatic search process of parallel and series connections. The difficulty in such approaches is to find a metrics being generic enough to allow mathematical analysis. The metrics presented here takes advantage of both activity measure and connections topology. This metrics proved to be consistent in both modeling (mathematical) and simulation (computational) worlds. This allows, for parallel and series connections, to induce a new conjecture. As far as we know, it is the first attempt of a method for the mathematical analysis of the learning simulation of basic structures (parallel and series) of components, at system level.

The manuscript is organized as follows. Section 2 presents the mathematical structures and activity-based metrics used. Section 3 situates the approach presented in the context of model engineering. Section 4 describes the simulation and mathematical results obtained for both parallel and series connections. Finally, a conclusion presents the new conjecture and draws the lines of the corresponding formal perspectives.

2 TOOLS

Discrete event system specification allows specifying locuses of activity (components) [9] and thus storing the amount of activity of components. The activity (efficiency) of components can be combined with the behavior

(performance) of compositions to find target solutions.

2.1 Discrete Event System Specification (*DEVS*)

The structure of both network and basic discrete event systems is presented here.

Definition 1. A *basic Discrete Event System Specification (DEVS)* is a structure:

$$DEVS = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

Where, X is the *set of input events*, Y is the *set of output events*, S is the *set of partial states*, $\delta_{ext} : Q \times X \to S$ is the *external transition function* with $Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$ the set of total states, $\delta_{int} : S \to S$ is the *internal transition function*, $\lambda : S \to Y$ is the *output function*, and $ta : S \to \mathbb{R}^{0,+}$ is the *time advance function*.

Definition 2. *A DEVS network* is a structure:

$$N = (X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select)$$

Where X is the *set of input events*, Y is the *set of output events*, D is the *set of component names*, for each $d \in D$, M_d is a basic model (whose structure differs from one *DEVS*-based formalism to another), for each $d \in D \cup \{N\}$, I_d is the *set of influencers* of d such that $I_d \subseteq D \cup \{N\}, d \notin I_d$ and for each $i \in I_d$: $Z_{i,d}$ is a *coupling function*, the $i-$ to $-d$ output translation, defined for: (i) *external input couplings*: $Z_{self,d} : X_{self} \to X_d$, with $self$ the *network name*, (ii) *internal couplings*: $Z_{i,j} : Y_i \to X_j$, and (iii) *external output couplings*: $Z_{d,self} : Y_d \to Y_{self}$, and $Select : 2^D - \{\emptyset\} \to D \cup \{\emptyset\}$ is the *sequential select function* (to select one component to execute its transition/output functions, among imminent components). Considering a set of components C candidate for internal transition, the sequential select function has constraint $Select(C) \in C \cup \{\emptyset\}$, i.e., only one component or no components can be selected among candidates.

2.2 Activity credit assignment (ACA)

Definition 3. *Event-based activity* $A_\xi(t' - t)$ [9] in an event set ξ consists of:

$$A_\xi(t' - t) = |\{ev_i = (t_i, v_i) \in \xi \mid t \leq t_i < t'\}|$$

Where t, t_i, t' are *time-stamps* and $v_i \in V$ is an *event value*.

Average event-based activity consists then of $\overline{A_\xi(t' - t)} = \frac{A_\xi(t' - t)}{t' - t}$.

For example, assuming the event trajectory depicted in Figure 1, the average event-based activity of the system corresponds to the following values for different time periods: $\overline{A_\xi(10)} = 0.3$, $\overline{A_\xi(20)} = 0.15$, $\overline{A_\xi(30)} \simeq 0.133$, $\overline{A_\xi(40)} = 0.175$.

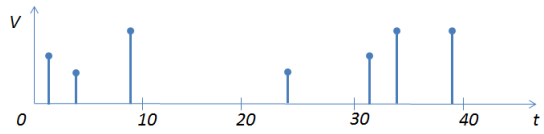

Figure 1: An example of event trajectory.

Definition 4. *Average external activity* $\overline{A_{ext}}$, related to the counting n_{ext} of *external transitions* $\delta_{ext}(s, e, x)$, over a time period $[t, t']$ consists of:

$$\begin{cases} s' \leftarrow \delta_{ext}(s, e, x) \Rightarrow n'_{ext} \leftarrow n_{ext} + 1 \\ \overline{A_{ext}(t' - t)} = \frac{n_{ext}}{t' - t} \end{cases}$$

Definition 5. *Average internal activity* $\overline{A_{int}}$, related to the counting n_{int} of *internal transitions* $\delta_{int}(s)$, over a time period $[t, t']$ consists of:

$$\begin{cases} s' \leftarrow \delta_{int}(s, e) \Rightarrow n'_{int} \leftarrow n_{int} + 1 \\ \overline{A_{int}(t' - t)} = \frac{n_{int}}{t' - t} \end{cases}$$

Definition 6. *Total average simulation activity* $\overline{A_s(t' - t)}$ is equal to:

$$\overline{A_s(t' - t)} = \overline{A_{ext}(t' - t)} + \overline{A_{int}(t' - t)}$$

$e_v : \Omega \times P \times P^* \to \mathbb{R}$ is the *evaluation function*, where $\rho^* \in P^*$ is the *target output segment* (a partial function $\rho^* : [t, t'] \to Y^*$) of the *target network name* (noted target network for short after) $k^* \in K$, $\rho \in P$ is the *output segment* (a partial function $\rho : [t, t'] \to Y$) of a *candidate network* $k \in K$, and $\omega \in \Omega$ is the *input segment* (a partial function $\omega : [t, t'] \to X$) of both *networks*. For each input segment $\omega \in \Omega$, the evaluation function e_v computes the *distance* between each *output segment* $\rho \in P$ (obtained by simulation) and the *target output segment* $\rho^* \in P^*$. Evaluation function compares simulation results between a *candidate network* $k \in K$ and a *target network* $k^* \in K$, i.e., $e_v(k, k^*)$ (noted $e_v(k)$ for short hereafter).

Definition 7. *A local optimum* $\widehat{k} \in K$ consists of $e_v(\widehat{k}) < e_v(k^*)$, where $k^* \in K$ is the global optimum (target network) and $e_v(k^*) = v^*$ with v^* the *maximum evaluation value*.

Definition 8. *At trial* $0 < r \leq R$ (with $r \in \mathbb{N}$), the *simulation credit* (*achievement*) of each component $i \in D$, over a simulation duration $[t, t']$, consists of $c_{i,r}(t' - t) = \overline{A_s(t' - t)} \times e_v(k)$, with $k \in K$.

Definition 9. Over a number of trials $R \in \mathbb{N}$, the *accumulated simulation credit* of a component $i \in D$ is the sum of accumulated credits at each *trial* $0 < r \leq R$: $c_{i,R}(t'-t) = \Sigma_{r=1}^{R} c_r(t'-t)$.

Definition 10. The *accumulated simulation credit* ratio of a component $i \in D$ consists of $\hat{c}_{s,i} = \frac{C_{s,i}}{C_{s,n}}$ where n is the index of the *last component* of a sequence of components $(DEVS_i)_{i=1}^{n}$ with $i \in D$ and $D \subset \mathbb{N}$.

Definition 11. The *accumulated predicted credit* of a component $i \in D$ consists of $c_{p,i} = \Sigma_{i \in D} P(i) ev(k)$, with $P(i)$ the *probability of activation of component* $i \in D$ and $k \in K$ a candidate network.

Definition 12. The *accumulated predicted credit ratio* of a component $i \in D$ consists of $\hat{c}_{p,i} = \frac{c_{p,i}}{c_{p,n}}$.

3 MODEL ENGINEERING

In [10], authors state that model reconstruction and reconfiguration is one of the important problems in model engineering: "According to the performance requirement change of model function caused by the diversity of demand and the environmental uncertainty, the model needs to be reconstructed and configured quickly. The model reconstruction is to adjust its internal structure without changing the main external features, in order to optimize the performance and make it easier to understand, maintain and transplant. The model configuration is to adjust and optimize the internal components and parameters without changing the basic structure of the model, so as to adapt the different requirements or changes in function and performance of the model. For the model engineering of the complex system, the model reconstruction and configuration management is a very important and challenging research."

Muzy and Zeigler [5] developed a framework for automatically rating component models and composing them according to the experimental frames in which they are placed. Components are assigned credit by correlating measures of their participation (activity) in simulation runs with run outcomes. These ratings are employed to bias component selection in subsequent compositions. Figure 2 sketches the position of the new decision and search process layers on the top of usual modeling and simulation processes. Model construction is biased via the synthesis from achievement components stored in repositories (or model-base).

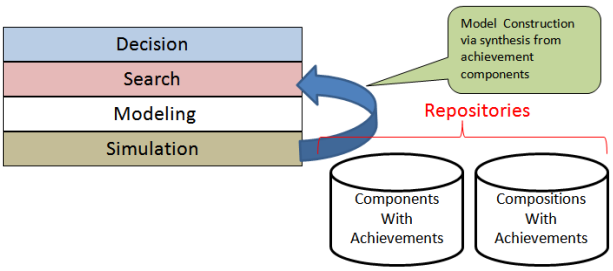

Figure 2: Decision and search in modeling and simulation.

Figure 3 presents the basic modeling and simulation entities used in activity-based credit assignment. A correlator entity is in charge of correlating performance evaluations obtained in an experimental frame and simulation results for credit assignment at component-based level.

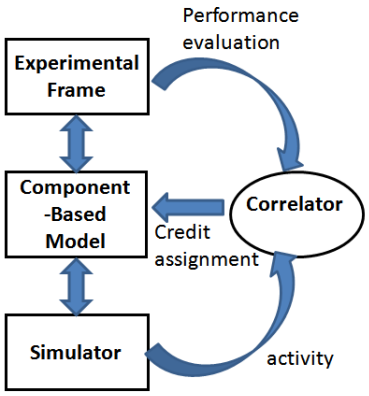

Figure 3: Basic modeling and simulation entities in activity-based credit assignment.

This activity-based approach to model construction also applies to model reconstruction and reconfiguration as defined by [10]. This is because the experimental frame represents the requirements of the model and when the latter changes, so should the frame representing them. A change in frame can then automatically induce a new search for composition of components to best meet the new requirements.

4 SERIES AND PARALLEL CONNECTIONS

In this set-up, the requirement for ACA to demonstrate is that it must converge to a stable assignment of credits to all steps where each right step has a credit that exceeds that of the corresponding wrong step. In this way, a composition strategy that selects from high valued component alternatives will select the correct composition. Assuming uniform independent random selection of alternatives.

We now proceed to show that this requirement holds using a stochastic analysis approach. We also verify the probability estimates obtained using direct simulation of the ACA as described in [6].

The *main assumptions* of the simulations are:

- At a given trial, for a right component, to be activated, all the precursors that are right have to be activated.
- At each step, a uniform random selection of components is achieved.
- Evaluations are deterministic. Each composition is evaluated once.
- Components contribute additively to the total outcome measure.
- Each wrong component has the same activity, when activated.
- Each right component has the same activity, when activated.

4.1 Series connections

Figure 4 represents the component-based implementation of a series connection, for four right steps. A *skill* is a learned sequence of actions. The problem here is to learn a skill that requires a sequence of actions. Assume that the set of base slots for actions and their coupling is known, the actions exist as alternative components for selection – the right component has to be selected for each slot. The sequence of activations is assumed to be known, and the actions exist as alternative components for selection – the right component has to be selected for each step. For simplicity, there are two actions, right and wrong, for each step. For example, "move an object from one place to another" consists of 1^{st} *step:* "move grabber to right place" vs. "move to another place"; 2^{nd} *step:* "lift the object" vs. "drop the object"; 3^{rd} *step:* "move the grabber to the target location", etc. Each step must be right before learning the next. For example, a skill has *4* steps. A coupled model is shown in Figure 4 that represents the components and couplings, where each component can be either *right* or *wrong*. Step *1* starts and if it is right, triggers step *2*; if step *1* is wrong, then the trial is over and a new candidate is generated. If step *2* is right, it triggers step *3*, etc.

Each right step sends an output of value *1* to be summed in the *Sum* component (*experimental frame*). Maximum score is then *4* if all steps are completed. A partial score of i is thus obtained for i steps completed

Figure 5 describes the accumulated predicted credits, the score, and the activation probability of right step components. Each right step accumulates its own score plus the sum of the *downstream* right step scores (since it participates in them). For example, for *2* right steps, with $\frac{1}{8}$ probability, the activation ends after right step *2*, and the score is then *2*.

Table 1 presents both predicted and simulation accumulated credits of each right step component. Looking at the table, it can be seen that for long enough runs (including many trials), for each right step component $i \in D$, the accumulated predicted credit ratio is equal to the accumulated simulation credit ratio $\hat{c}_{p,i} = \hat{c}_{s,i}$.

Right step	$\hat{c}_{p,i}$	$c_{s,i}$	$\hat{c}_{s,i}$
1	3.75	0.30	3.75
2	2.75	0.22	2.75
3	1.75	0.14	1.75
4	1	0.08	1

Table 1: Analysis of accumulated credit for each right step.

Figure 6 describes the *accumulated credits*, the *score*, and the *activation probability* of *wrong step* components. Each wrong step accumulates the previous step score. For example, for wrong step *3*, with $\frac{1}{8}$ probability, the activation ends after *right step 2*, and the *score* is then *2*.

As for right steps, Table 2 represents the *predicted* and *simulated accumulated credits* of each *wrong step* component. This table tells us that for each wrong step component $i \in D$, the accumulated predicted credit ratio is equal to the accumulated simulation credit ratio $\hat{c}_{p,i} = \hat{c}_{s,i}$. Also, when comparing Tables 1 and 2, it can be observed that the wrong step component with the highest accumulated credit $c_{s,i}$ still gets a lower accumulated credit than the right step component with the lowest accumulated credit. This should ensure a good ranking of all the components of the candidate set.

Wrong step	$\hat{c}_{p,i}$	$c_{s,i}$	$\hat{c}_{s,i}$
1	0	0	0
2	1.33	0.04	1.33
3	1.33	0.04	1.33
4	1	0.03	1

Table 2: Analysis of accumulated credit for each wrong step, with $\hat{c}_{s,i} = \hat{c}_{p,i}$.

The stochastic analysis generalizes to a series coupling of any length. For convenience let's consider such a connection with infinite length. Then at any step number, N, a wrong step gets credit for the previous $N-1$ steps and

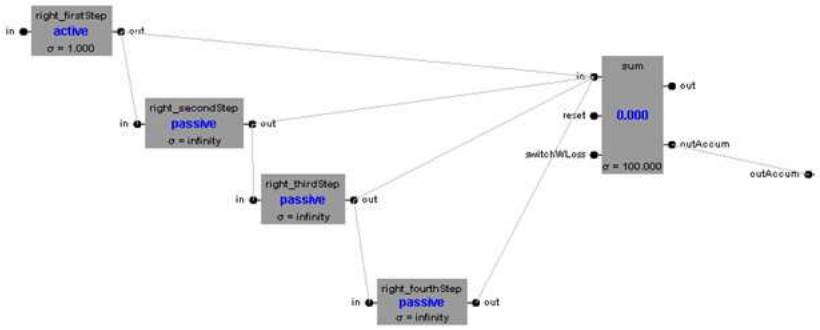

Figure 4: Series connection. *Step 1* initiates action – it needs to be right to activate *step 2* and to add *1* to the sum. *Step 2* continues action – it needs to be right to activate *step 3* and to add *1* to the sum – and so on.

this occurs with proability $P_F(N)$ given by a geometric (distribution) first failure process. A right step receives this credit plus credit for any subsequent right steps until a wrong step, for which the probabability distribution is a geometric first failure process. Thus each right step has a credit that exceeds that of the corresponding wrong step by an expected amount given by the mean of the geometric first failure distribution.

4.2 Parallel connections

Here the simulation consists of learning a skill that requires concurrent execution of actions. Assume that the set of base slots for actions is known, and the actions exist as alternative components for selection – the right component has to be selected for each slot. For simplicity, assume there are two actions, right and wrong, for each slot. For example: Move right arm forward and left leg backward simultaneously. You must get each slot right to get the whole skill right, but you get partial reward for getting one slot right while the other is wrong.

Figure 7 represents the component-based implementation of the parallel case, for four right steps. The skill has *4* slots. A coupled model is shown next that represents the components and coupling, where each component can be either right or wrong. Step *1* is set to be right. When it starts triggering the other slots. Each right slot sends an output to the Sum (experimental frame).

In such a parallel composition, with every component uniformly and independently randomly selected as either right or wrong, a Bernoulli distribution determines the number of right and wrong selected components at each trial. Focusing on one component, the distribution for other components is the same whether it is selected as right or wrong. The only difference being that a right selection is active and therefore receives one more unit of credit than the corresponding wrong alternative.

Table 3 represents the *accumulated predicted credit* of each *right step* $r \in D$, $c_{p,r}$, the *accumulated predicted credits* of each *wrong step* $w \in D$, $c_{p,w}$, and the ratios of total accumulated credits of right slots to wrong slots defined as predicted $\tilde{c}_p = \frac{\Sigma_{r \in D} c_{p,r}}{\Sigma_{w \in D} c_{p,w}}$ and simulated $\tilde{c}_s = \frac{\Sigma_{r \in D} c_{s,r}}{\Sigma_{w \in D} c_{s,w}}$.

# **right**	$P(i)$	$c_{p,w}$	$c_{p,r}$
0	.25	1	2
1	.5	1.5	2.5
2	.25	2	3
Total		4.5	7.5
$\tilde{c}_p$			7.5/4.5=1.6
$\tilde{c}_s$			12/7.0=1.7

Table 3: Analysis of accumulated credits for each right and wrong step.

Table 3 suggests that both predicted and simulated ratios of total accumulated credits of right slots to wrong slots are approximatively equal $\tilde{c}_p \simeq \tilde{c}_s$. This should ensure a good ranking of right and wrong components for simulation-based learning of target parallel connections.

5 CONCLUSION AND PERSPECTIVES

In the context of sequential machines [3], for series and parallel connections, accumulated credit seems to be the right metrics of the set of candidate network solutions.

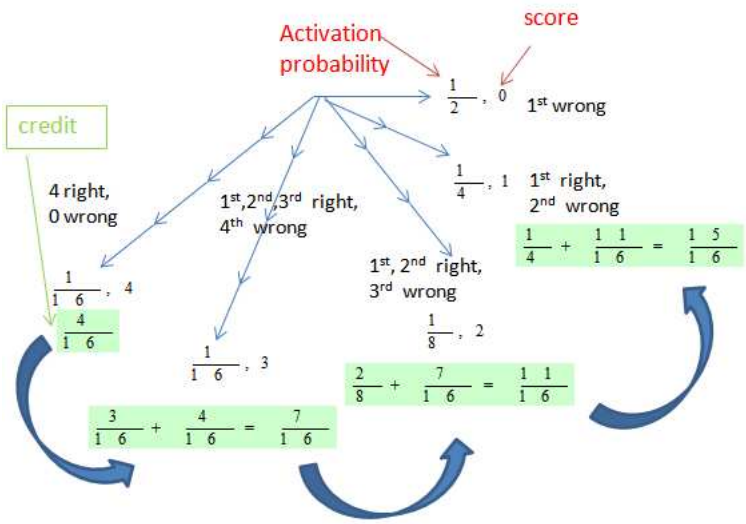

Figure 5: Analysis of accumulated credit for each right step – component in the target set.

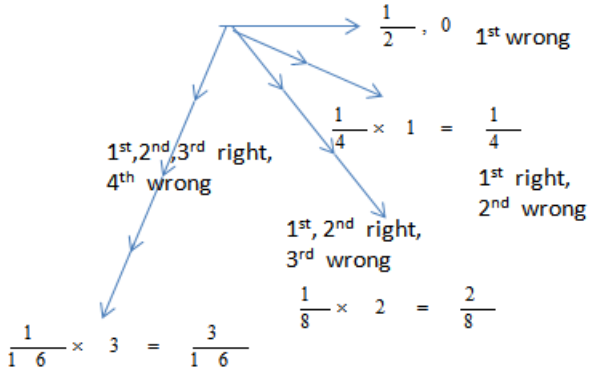

Figure 6: Analysis of accumulated credit for each wrong step – component not in the target set.

Based on the results obtained here, the following conjecture can be derived:

Conjecture 1. *Consider any composition whether series, parallel, or feedback in any combination. Let each component have right and wrong alternatives and let the outcome be measured by the number of right selections. If the alternatives are selected uniformly and independently at random in every trial, then the expected credit assigned by ACA to each right alternative always exceeds the expected credit assigned to the corresponding wrong alternative.*

Once this conjecture has been proved, i.e., that connection candidate sets are well ordered according to their achievements, it would be possible to correlate learning times (to find components) to the amount of activity required. It should be that the more activity invested in the learning process the smaller the learning time. More generally, the automatic learning of system structures, as demonstrated here through parallel and series connections, constitutes a formal basis for model reconstruction and reconfiguration in model engineering as defined by references [10, 5]. In [5], ACA proved to be more efficient than a random model-base approach (selecting randomly components in a repository), for a model embedding both parallel and series chains of components and possibly feed-back loops. Testing the conjecture obtained here in the framework presented in [5], constitutes a challenging perspective for ACA mathematical analysis of more complex systems.

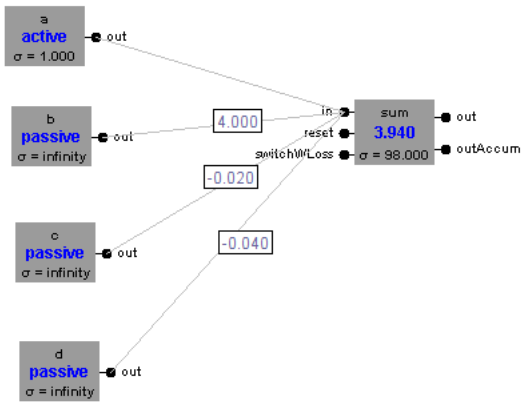

Figure 7: Parallel connections.

REFERENCES

[1] Hervé Gallaire and Michael A. Harrison. Decomposition of linear sequential machines. *Mathematical Systems Theory*, 3(3):246–287, 1969.

[2] Hervé Gallaire. Decomposition of linear sequential machines. ii. *Mathematical Systems Theory*, 4(2):168–190, 1970.

[3] Juris Hartmanis. *Algebraic Structure Theory of Sequential Machines (Prentice-Hall International Series in Applied Mathematics)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1966.

[4] Xiaolin Hu and Bernard P. Zeigler. Linking information and energy - activity-based energy-aware information processing. *Simulation*, 89(4):435–450, 2013.

[5] Alexandre Muzy and Bernard P. Zeigler. Activity-based credit assignment (aca) heuristic for simulation-based stochastic search in a hierarchical model-base of systems. *Accepted for publication in IEEE Systems Journal*, 2014.

[6] Alexandre Muzy and Bernard P. Zeigler. Activity-based credit assignment (aca) in hierarchical simulation. In *SpringSim (TMS-DEVS)*, page 5, 2012.

[7] George J. Klir and Doug Elias. *Architecture of Systems Problem Solving*. Springer, 2003.

[8] A.W. Wymore. *Model-Based Systems Engineering*. Systems Engineering. Taylor & Francis, 1993.

[9] Alexandre Muzy, Luc Touraille, Hans Vangheluwe, Olivier Michel, Mamadou Kaba Traoré, and David R. C. Hill. Activity regions for the specication of discrete event systems. In *Spring Simulation Multi-Conference Symposium On Theory of Modeling and Simulation (DEVS)*, pages 176–182, 2010.

[10] Lin Zhang, Yuewei Shen, Xuesong Zhang, Xiao Song, Fei Tao, and Ying Liu. The model engineering for complex system simulation. In *26th European Modeling and Simulation Symposium (EMSS) - Model Engineering Workshop, accepted for publication*, 2014.